(2)

PI Names: Krithi Ramamritham, John A. Stankovic
PI Institution: University of Massachusetts
PI Phone Number: 413 545-0196 or 413 545-0720
PI E-mail Address: krithi@.cs.umass.edu, stankovic@cs.umass.edu
Grant or Contract Title: Predictable and Adaptable Complex Real-Time Systems
Grant or Contract Number: N00014-92-J-1048
Reporting Period: 1 Oct 91 - 30 Sep 93

1. Productivity Measures:

Refereed papers submitted but not yet published:	8
Refereed papers published:	13
Unrefereed reports and articles:	12
Books or parts thereof submitted but not yet published:	0
Books or parts thereof published:	7
Patents filed but not yet granted:	0
Patents granted:	0
Invited presentations:	22
Contributed presentations:	14
Honors received:	
Technical Society Appointments	2
Conference Committee Roles	50
Editorships	5
IEEE Fellow	1
Prizes or awards received:	
Promotions obtained:	1
Graduate students supported >=25% of full time:	7
Post-docs supported >=25% of full time:	0
Minorities supported (include Blacks, Hispanics, American Indians and other native Americans such as Aleuts, Pacific Islanders, etc., Asians, and Indians):	4

PI Names: Krithi Ramamritham, John A. Stankovic
PI Institution: University of Massachusetts
PI Phone Number: 413 545-0196 or 413 545-0720
PI E-mail Address: krithi@nirvan.cs.umass.edu, stankovic@cs.umass.edu
Grant or Contract Title: Predictable and Adaptable Complex Real-Time Systems
Grant or Contract Number: N00014-92-J-1048
Reporting Period: 1 Oct 91 - 30 Sep 93

2. Summary of Technical Progress

Our investigations covered many aspects of real-time system research and development with special emphasis on scheduling algorithm development, evaluation, and *realization*. Results were also obtained in real-time operating systems, fault-tolerance, software tools, hardware/software integration and real-time databases.

We have also attempted to test the concepts and technologies developed by incorporating them in a flexible manufacturing testbed.

PI Names: Krithi Ramamritham, John A. Stankovic
PI Institution: University of Massachusetts
PI Phone Number: 413 545-0196 or 413 545-0720
PI E-mail Address: krithi@nirvan.cs.umass.edu, stankovic@cs.umass.edu
Grant or Contract Title: Predictable and Adaptable Complex Real-Time Systems
Grant or Contract Number: N00014-92-J-1048
Reporting Period: 1 Oct 91 - 30 Sep 93

3. Detailed Summary of Technical Results

We made research progress on four fronts: (1) scheduling in complex, dependable, real-time systems, (2) developing software support tools, most importantly, mapping schemes for deriving task-level descriptions from process-level descriptions, (3) development of our operating system kernel, (4) design and implementation of architecture support for dynamic real-time scheduling and (5) real-time databases.

We now provide a brief summary of our new work in these areas.

3.1 Scheduling in Complex Real-Time Systems

a) Many distributed real-time applications involve complex periodic activities with periods smaller than the end-to-end timing constraints, often due to tight data arrival requirements. That is, a new instance of a periodic activity will come into existence before the previous instance has been completed. Also, such activities typically involve communicating modules where some modules may be replicated for resiliency. For such activities, pipelined execution allows us to meet the various resource and timing constraints imposed on them. We have developed an approach to statically scheduling a pipelined execution of a set of periodic activities that have the above characteristics. Effectiveness of the approach has also been shown via simulation studies.

b) In addition to periodic activities, most real-time systems also have activities that arrive dynamically. The system must be responsive to these activities while maintaining the guaranteed execution of periodic activities. We present a simple technique that takes a static schedule – that specifies scheduled start times – into a more flexible schedule – that specifies execution windows – for the components of the activities. This flexibility is shown to enhance the responsiveness to aperiodic arrivals.

c) We have developed a robust earliest deadline scheduling algorithm for dealing with sporadic tasks under overloads in a hard real-time environment. The algorithm synergistically combines many features including a very important minimum level of guarantee, dynamic guarantees, graceful degradation in overloads, deadline tolerance, resource reclaiming, and dynamic re-guarantees. A necessary and sufficient schedulability test is presented, and an

efficient $O(n)$ guarantee algorithm is proposed. The new algorithm is evaluated via simulation and compared to several baseline algorithms. The experimental results show excellent performance of the new algorithm in normal and overload conditions.

d) We have produced a formal scheduling result that is based on the concept of quasi-normality. This allows the integration of precedence constraints, resource requirements and deadlines. The formal results can be applied to the well known priority ceiling or stack based resource scheduling algorithms, extending that work (in a formal and practical way) to handle precedence constraints.

e) Many real-time systems have both performance requirements and reliability requirements. Performance is usually measured in terms of success in completing tasks on time. Reliability is determined by hardware and software failure models. In many situations, there are tradeoffs between task performance and task reliability. Thus, a mathematical assessment of performance-reliability tradeoffs is necessary to evaluate the performance of real-time fault-tolerance systems. Assuming that reliability of task execution is achieved through task replication, we have developed an approach to mathematically determine the replication factor for tasks. The goal is to maximize the total performance index, which is a performance-related reliability measurement. The technique is based on a continuous task model and we have shown that it very closely approximates discrete models and tasks with varying characteristics.

f) We have developed the notion of a FERT (Fault Tolerant Real-Time Entity) that allows one to specify flexible redundancy management subject to strict time constraints. Details of the notation and how it interacts with the run time system, most importantly, the scheduling, have been worked out. Analysis and implementation are still required. This work is based on strongly emphasizing the concept of *reflection* in system design and implementation. See also the Kernel description below.

g) We have developed Well-Timed Scheduling, a framework for on-line scheduling algorithms. To predict whether a task will complete by its deadline, schedulability analysis has to be carried out. The quality of the analysis, as well as its computational overheads, depends on ***when*** the analysis is performed and ***how many*** tasks are involved. We have developed an analytically based approach to schedulability analysis, called *Well-Timed Scheduling*, that presents a ***framework*** for on-line scheduling algorithms. This framework provides a methodical approach to quantifiable guarantees of timing constraints with potentially low scheduling overheads. Using this approach, the tasks are scheduled at an "opportune" time, rather than at arrival or at dispatch time. The analytical derivation of the "opportune" time is based on recent theoretical results, and it is validated through simulation. Aside from run-time benefits (e.g., low scheduling overheads), Well-Timed Scheduling is useful as a design tool. It can, for example, be used to determine the number of processors needed to achieve the required level of system's guaranteed performance, for a given $M/G/c$ system with timing constraints. Another advantage of Well-Timed-Scheduling is that it lends itself to use with different scheduling policies, including the one based on heuristics that has been used in Spring for a number of years.

h) We have completed the analysis of the performance of our heuristic algorithm, called H. For pragmatic reasons, heuristic algorithms which integrate CPU and resource requirements, are required for on-line, hard real-time task scheduling. For algorithms that schedule hard real-time tasks, both the ability to generate feasible schedules and the quality of the generated feasible schedules, expressed in terms of the schedule length, are important performance metrics. Whereas their ability to find feasible schedules can be determined using simulation studies, only analysis can reveal their performance with respect to the schedule length bound.

As a result of the analysis, we have developed an algorithm H_k, which tries to keep at least k processors busy, if possible. In fact, H_k combines the features of two known heuristic scheduling algorithms: *list scheduling* and *the H scheduling algorithm.* We analyze its schedule length bound for both uniform tasks, i.e., tasks with the same computation time and non-uniform tasks, i.e., tasks with the arbitrary computation times. When $k = 2$, the time complexity of H_k is the same as the complexity of the H scheduling algorithm and list scheduling, which is $O(n^2r)$, where n is the number of tasks and r is the number of r esources. Whereas the H scheduling algorithm has a poor schedule length bound but performs very well in finding feasible schedules, and list scheduling does not perform well in finding feasible schedules but has a good bound, our results shows that H_2 has both a good schedule length bound and performs well in finding feasible schedules.

i) Finally, in collaboration with NYU/UT we have considered the problem of preemptively scheduling sporadic task requests in both uni- and multi-processor environments. Specifically, with respect to on-line scheduling algorithms that must direct the service of sporadic task requests we have quantified the benefit of **clairvoyancy**, i.e., the power of possessing knowledge of various task parameters of future events. If a task request is successfully scheduled to completion, a value equal to the task's execution time is obtained; otherwise no value is obtained. We have proved that no on-line scheduling algorithm can guarantee a cumulative value greater than 1/4th the value obtainable by a clairvoyant scheduler; i.e., we prove a 1/4th upper bound on the **competitive factor** of on-line real-time schedulers. We have also shown an on-line uniprocessor scheduling algorithm TD_1 that actually has a competitive factor of 1/4; this bound is thus shown to be tight. We have also considered the effect of restricting the amount of overloading permitted (the "loading factor"), and have quantified the relationship between the loading factor and the upper bound on the competitive factor. Other results of a similar nature deal with the effect of value densities (measuring the "importance" or "type" of a task). Generalizations to dual-processor on-line scheduling are also considered. For the dual-processor case, we have proved an upper bound of 1/2 on the competitive factor. This bound is shown to be tight in the special case when all the tasks have the same density and zero laxity.

3.2 Software Support in Spring: Mapping Program Representations to Task Representations

The Spring system uses a scheduler that constructs explicit plans for executing application programs to ensure that timing constraints are satisfied. Such scheduling requires a

detailed representation of the worst case run-time behavior of the application. Specifically, our scheduler uses the behavior of computations represented as a group of one or more tasks whose execution order is constrained by precedence relations. Each task has a worst case execution time (WCET), and uses a subset of the resources available on the system. Constructing such a behavioral description requires detailed information about the properties and requirements of the application, system, and target hardware.

The use of such a run-time representation is in stark contrast to the familiar programming model describing computations in terms of processes, or groups of processes, where a process is a single thread of control in an independent address space. Most systems use essentially the same representation at run-time. Proper run-time management in a real-time system depends on accurately *predicting* program execution behavior, which must be based on information either specified in the program source or derived from it during translation. We have developed a method for deriving behavioral predictions while translating between the programming and run-time representations used by the Spring system. A graph representation of the program is derived from the intermediate representation the compiler uses to emit code. This graph is then reduced and analyzed to make behavioral predictions.

We have developed a method for deriving behavioral predictions while translating between the programming and run-time representations used by the Spring system. A graph representation of the program is derived from the intermediate representation the compiler uses to emit code. This graph is then reduced and analyzed to make behavioral predictions. The end result is a smart compiler for real-time systems.

The Spring system description language (SDL), plays an important role in both its source and compiled forms. SDL source files provide a way for developers to specify the properties of all parts of the system in great detail. When compiled, this information is available for use by other tools needing to use, modify, or add to it. The SDL provides vital support for specification, compilation, and execution of applications on the Spring system, as well as for specification of simulations run under Spring's scheduling testbed.

The usefulness of SDL was put to test when an existing robotics process for assembling printed circuit boards was coded using SDL structures so that the resulting code could be run on Spring. Also, SDL in conjunction with Spring, forms the base upon which a demonstration application has been developed in the context of flexible manufacturing using dextrous robot hands.

3.3 The Spring Real-Time Operating System Kernel

The Spring kernel stresses the real–time and flexibility requirements, and also contains several features to support fault tolerance.

During this contract period, we have worked the development of the Spring hardware testbed by focusing on its distribution. Nodes in the Spring distributed system are connected via Ethernet (for non-real-time traffic) as well as via a predictable replicated memory based on SCRAMNET (developed by SYSTRAN) – a fiber optic register insertion ring (for real-

time traffic). SCRAMNET is a replicated shared memory architecture that uses a fiber optic register insertion ring, running at 150 MBits/sec. Each node has 2 MB of shared common memory, and writes to this memory are broadcast (circulated) about the ring. The fiber optic medium has bit error rates of around 10^{-11}.

Based on SCRAMNET, we have completed designing the real-time interprocess communication (IPC) in the Spring Kernel. It is designed to offer *predictable* communication. Thus, the communication primitives have bounded execution time and synchronous communication does not introduce unpredictable blocking. We achieve this by tight integration of IPC with Spring scheduling and through compile-time support that maps processes to schedulable tasks.

The Spring hardware testbed is based on conventional target hardware. This created severe constraints on its design and performance. In spite of this, we have ensured the predictability of the resulting system. We believe our experience provides specific lessons for the architectural support of real-time operating systems. In this context, we have examined to what extent we can make predictions about the effects of caches on the system, and the extent to which these predictions can be used to reduce the pessimism of WCET estimates while preserving their correctness.

We also worked on Spring's ability to handle more complicated task structures, especially precedence relationships and external processes, for robotic computations. We also established an overriding principle for the kernel and its extensions based on *reflection*. We are attempting to integrate more and more fault tolerance features into the kernel.

3.4 Architecture Support

We have just completed the design and fabrication of a scheduling coprocessor to reduce the overheads of dynamic scheduling. This novel co-processor for multi-processor scheduling employs a parallel VLSI architecture for scheduling. It can be scaled for different numbers of tasks, resources, and internal word lengths. The implementation uses an advanced clocking scheme to allow further scaling using future IC technologies. The interface to the host system, and its implications for scheduler performance, are discussed. With an internal clock rate of up to 100MHz, we expect the current design to speed up the portion of the scheduling operation it implements by over three orders of magnitude, speeding up the scheduling operation as a whole by 30 fold, and thus significantly altering a major bottleneck in real-time systems which perform dynamic execution planning. Subsequent versions of the co-processor will implement more of the scheduling operation in hardware, continuing to improve scheduling performance.

3.5 Real-Time Databases

Real-time transaction processing requires an integrated set of protocols that must not only satisfy database consistency requirements but also operate under timing constraints.

In work conducted as part of the first grant listed above, we developed, implemented, and evaluated integrated suites of algorithms that support real-time transaction processing and deal with the following issues: time-constrained cpu scheduling, concurrency control (based on locking and on optimistic concurrency control), conflict resolution, transaction restart, transaction wakeup, deadlock, buffer management, and disk I/O scheduling. The algorithms directly address real-time constraints. The implementation was performed on a testbed called Real-Time Concurrency And Recovery Algorithm Testbed (RT-CARAT) which contains all the major features of a transaction processing system. Some of the salient results include: (1) real-time cpu scheduling, conflict resolution, and disk I/O scheduling are the three main factors in achieving good performance, (2) various conflict resolution protocols which directly address deadlines and criticalness can have a important impact on performance over protocols that ignore such information, (3) deadlock resolution and transaction restart policies tailored to real-time constraints seem to have negligible impact on overall performance, (4) optimistic concurrency control outperforms locking except when data contention is high, (5) real-time buffer management does not provide significant gain over typical buffer management techniques when the database is supported by local and global buffers, and (6) our new disk I/O scheduling algorithms are much more effective than others currently available.

We also showed that while priority inheritance is not effective in real-time databases, a new protocol that we developed called conditional priority inheritance has been shown to be very effective. It takes into account the amount of work yet to be done by a transaction to decide whether a transaction must inherit priority from a higher priority waiting transaction or must abort. This is an important result because priority inheritance is so widely used at the OS level where critical sections are short, but we showed that a straightforward use of it at the database level does not work, and further, we developed a technique that does work.

PI Names: Krithi Ramamritham, John A. Stankovic
PI Institution: University of Massachusetts
PI Phone Number: 413 545-0196 or 413 545-0720
PI E-mail Address: krithi@nirvan.cs.umass.edu, stankovic@cs.umass.edu
Grant or Contract Title: Predictable and Adaptable Complex Real-Time Systems
Grant or Contract Number: N00014-92-J-1048
Reporting Period: 1 Oct 91 - 30 Sep 93

4. Publications, Presentations, Reports and Honors:

4.1 Publications:

- *Advances in Hard Real-Time Systems*, IEEE Computer Society Press, Washington, DC, September 1993, J. Stankovic and K. Ramamritham.
- Distributed Computing, invited paper, *Encyclopedia of Telecommunications*, Marcel Dekker, Inc., N.Y., pp. 289–319, 1993; also in *Readings in Distributed Computing Systems*, IEEE Press, J. Stankovic.
- Resource Reclaiming in Multiprocessor Real-Time Systems, *IEEE Transactions on Parallel and Distributed Systems*, Volume 4, Number 4, April 1993, pp. 382-97, C. Shen, K. Ramamritham and J. Stankovic.
- Real-Time Databases, invited paper, *Journal of Distributed and Parallel Databases*, Volume 1, Number 2, 1993, pp. 199- 226, K. Ramamritham.
- On Using Priority Inheritance in Real-Time Databases, *Real-Time Systems Journal*, Vol 4, No. 3, pp 243-68, 1992, J. Huang, J. Stankovic, K. Ramamritham, D. Towsley and B. Purimetla.
- The Spring Scheduling Co-Processor: Design, Use and Performance, *Real-Time Systems Symposium*, D. Niehaus, K. Ramamritham, J. Stankovic, G. Wallace, C. Weems, W.Burleson and J. Ko, Dec. 1993.
- The Spring Scheduling Co-Processor: A Scheduling Accelerator, *International Conference on Computer Design*, IEEE, November 1993, W.Burleson, J. Ko, D. Niehaus, K. Ramamritham, J. Stankovic, G. Wallace and C. Weems.
- Scheduling Algorithms and Operating Systems Support for Real-Time Systems, accepted for *Proceedings of the IEEE*, to appear, K. Ramamritham and J. Stankovic.
- Editorial: Resource Allocation in Real-Time Systems, *Real-Time Systems Journal*, Vol. 5, No. 2-3, pp. i-vi, May 1993, J. Stankovic.

- How to Integrate Precedence Constraints and Shared Resources in Real-Time Scheduling, to appear, *IEEE Transactions on Computers*, preliminary version appeared in *IEEE Workshop on Real-Time Operating Systems and Software*, May 1993, M. Spuri and J. Stankovic.

- RED: A Robust Earliest Deadline Scheduling Algorithm, submitted to *IEEE Transactions on Computers*, Feb. 1993, (and short form accepted for *Third International Workshop on Responsive Computer Systems*, Sept. 1993), G. Buttazzo and J. Stankovic.

- Implications of Classical Scheduling Results For Real-Time Systems, submitted to *IEEE Computer*, March 1993, J. Stankovic, M. Spuri, M. Di Natale, and G. Buttazzo.

- Adaptive Fault Tolerance for Real-Time Systems, *Third International Workshop on Responsive Computer Systems*, longer version *PDCS*, Sept. 1993, A. Bondavalli, J. Stankovic, and L. Strigini.

- Major Real-Time Challenges for Mechatronic Systems, invited paper, *International Workshop on Mechatronical Computer Systems for Perception and Action*, June 1993, J. Stankovic.

- Reflective Real-Time Systems, submitted to *Fourth IFIP Conference on Dependable Computing for Critical Applications*, June 1993, J. Stankovic.

- Editorial: Real-Time Kernel Interfaces, *Real-Time Systems Journal*, Vol. 5, No. 1, pp. 5-8, March 1993, J. Stankovic.

- Real-Time Operating Systems, invited paper, *Proc. NATO Advanced Study Institute on Real-Time Computing*, to appear, J. Stankovic.

- Real Time Programming, W.A. Halang and K. Ramamritham, Editors, Pergamon Press, 1992.

- Scheduling in Real-Time Transaction Systems, in Foundations of Real-Time Computing Scheduling and Resource Management, Andre van Tilborg and Gary Koob, Editors, pp. 157-84, 1992, J.A. Stankovic, K. Ramamritham, D. Towsley.

- Scheduling Strategies Adopted in Spring: An Overview, in Foundations of Real-Time Computing: Scheduling and Resource Management, Andre van Tilborg and Gary Koob, Editors, pp. 277-305, 1992, K. Ramamritham, J.A. Stankovic.

- The Spring Kernel: A New Paradigm for Hard Real-Time Operating Systems, in Operating Systems for Mission Critical Computing, Agrawala, Gordon and Hwang, Editors, IOS Press, Inc., Amsterdam, pp. 86-117, 1992, J.A. Stankovic and K. Ramamritham.

- On Using Priority Inheritance in Real-Time Databases, *Real-Time Systems Symposium*, Dec. 1991, J. Huang, J.A. Stankovic, K. Ramamritham, D. Towsley.

- Real-Time Computing, chapter in Encyclopedia of Computer Science and Technology, Marcel Dekker, Inc., NY, NY, Vol. 23, Supplement 8, pp. 335-51, 1992, J.A. Stankovic.
- Real-time Computing, *BYTE*, invited keynote paper, pp. 155-60, August, 1992, J.A. Stankovic.
- Distributed Real-Time Computing: The Next Generation, *Special issue of Journal of the Society of Instrument and Control Engineers of Japan*, invited keynote paper, Vol. 31, No. 7, pp. 726-36, 1992, J.A. Stankovic.
- Performance Evaluation of Two New Disk Scheduling Algorithms for Real-Time Systems, *Real-Time Systems Journal*, Vol. 3, No. 3, pp. 307-36, September 1991, S. Chen, J.A. Stankovic, J. Kurose, D. Towsley.

4.2 Presentations:

- How to Integrate Precedence Constraints and Shared Resources in Real-Time Scheduling, *Real-Time Operating Systems and Software Workshop*, May 1993, M. Spuri and J. Stankovic.
- Major Real-Time Challenges for Mechatronical Systems, invited speaker, *International Workshop on Mechatronical Computer Systems for Perception and Action*, June 1993, J. Stankovic, Halmsted, Sweden. (Elected workshop discussion leader).
- Network Services Database - A Distributed Active Real-Time Database, *First IEEE Workshop on Real-Time Applications*, May 1993, R. Sivasankaran, B. Purimetla, J. Stankovic, K. Ramamritham.
- A Study of Distributed Real-Time Active Database Applications, *Workshop on Parallel and Distributed Real-Time Systems*, April 1993, B. Purimetla, R. Sivasankaran, J. Stankovic, K. Ramamritham and D. Towsley.
- SpringNet: An Architecture for High Performance Distributed Real-Time Computing, *Workshop on Parallel and Distributed Real-Time Systems*, April 1993, J. Stankovic, D. Niehaus and K. Ramamritham.
- An Integrated Real-Time Data Management Architecture for Industrial Control Systems, abstract only, *IEEE Workshop on Parallel and Distributed Real-Time Systems*, April 1993, J. Huang and J. Stankovic.
- The Integration of Scheduling and Fault Tolerance in Real-Time Systems, *Workshop on Imprecise Computation*, December 1992, J. Stankovic and F. Wang.
- Bounds on the Schedule Length of Some Heuristic Scheduling Algorithms for Hard Real-Time Tasks, *Real-Time Systems Symposium* , December 1992, F. Wang, K. Ramamritham and J. Stankovic.

- Multiprocessing and Distributed Scheduling Results and Architectural Support, *Proceedings Responsive Systems Workshop*, Oct. 1991, J.A. Stankovic, K. Ramamritham.
- Real-Time Operating Systems: What's Wrong With Today's Systems and Research Issues, *Proc. Real-Time OS Workshop*, May 1992, J.A. Stankovic.
- Scheduling in Dynamic Real-Time Systems, *Proc. Real-Time OS Workshop*, May 1992, K. Ramamritham.
- SpringNet: A Scalable Architecture for High Performance, Predictable, and Distributed Real-Time Computing, J.A. Stankovic, D. Niehaus, K. Ramamritham, *Proc. Workshop on Architectural Aspects of Real-Time Systems*, extended ab stract, Dec. 1991.
- Heuristic Algorithms for Multiprocessor Scheduling of Hard Real-time Tasks, F. Wang, K. Ramamritham, J. Stankovic, *Real-Time Systems Symposium*, extended abstract, Dec. 1991.
- On Using Priority Inheritance in Real-Time Databases, J. Huang, J.A. Stankovic, K. Ramamritham, D. Towsley, *Real-Time Systems Symposium*, Dec. 1991.

4.2.1 Talks and Colloquia

- Seminar, Technical University of Vienna, May 5, 1993, Stankovic.
- Seminar, CS Dept., University of Genoa, March 4, 1993, Stankovic.
- Seminar, CS Dept., University of Pisa, Feb. 19, 1993, Stankovic.
- PDCS Workshop Speaker (Stankovic), Italy, Jan. 7, 1993.
- CS Seminar, Univ. of Torino, Nov. 18, 1992, Stankovic.
- CS Seminar, Univ. of Milan, Nov. 17, 1992, Stankovic.
- NAT0 Workshop, St. Martin, Oct. 6-9, 1992, Invited Speaker, Panel Chair on Real-Time Operating Systems, and Panel Member on Real-Time Databases, Stankovic.
- Seminar, Instituto Di Elaborazione Della Informazione, Sept. 30, 1992, Stankovic.
- Seminar, Scuola Superiore Santa Anna, Pisa, Italy, Sept. 16, 1992, Stankovic.
- Speaker, Workshop on Object Oriented Support for Fault Tolerance, Blanchland, England, April 1993, Stankovic.
- Mead Data Central, April 1993, Ramamritham.
- Boston University, Boston, MA, March 1993, Ramamritham.

- Indian Institute of Science, Bangalore, India, Dec 1992, Ramamritham.
- Texas A&M University, College Station, TX, Oct 1992, Ramamritham.
- University of Houston, Houston, TX, Oct 1992, Ramamritham.
- Georgia Institute of Technology, Atlanta, GA, Oct 1992, Ramamritham.
- Invited speaker at the Symposium on Reliable Distributed Systems, Princeton, NJ, October, 1993, Ramamritham.
- Tutorial speaker on Real-Time Database Systems at the Conference on Management of Data, Bangalore, India, Dec 1992, Ramamritham.
- Invited speaker at the Conference on Management of Data, Bangalore, India, Dec 1992, Ramamritham.
- Tutorial speaker on Real-Time Database Systems at SIGMOD 93, Ramamritham.
- Taught a week-long course on real-time systems at Alcatel, Vienna, April, 1993, Ramamritham.

4.3 Reports:

- CMPSCI TR93-01, The Spring System Description Language, D. Niehaus, J. Stankovic, and K. Ramamritham.
- CMPSCI TR93-51, Well-Timed Scheduling and Scheduling with Precedence Constraints, PhD Thesis, Goran Zlokapa.
- CMPSCI TR93-56, Reflective Real-Time Systems, Stankovic.
- COINS TR92-60, Distributed Deadlock Detection and its Application to Real-Time Systems, PhD Thesis, Chia-Shiang Shih.
- COINS TR92-65, An Integrated Approach to Dynamic Task and Resource Management in Multiprocessor Real-Time Systems, PhD Thesis, Chia Shen.
- COINS TR91-74, SpringNet: A Scalable Architecture for High Performance, Predictable, and Distributed Real-Time Computing, J.A. Stankovic, D. Niehaus, K. Ramamritham, (Submitted to 12th Intl. Conf. on DCS).
- COINS TR91-75, Program Representation and Translation for Predictable Real-Time Systems, D. Niehaus.
- COINS TR92-01, Distributed Real-time Computing: The Next Generation, J.A. Stankovic.

- COINS TR92-40, Real-Time Operating Systems, J.A. Stankovic. (Invited short article for Encyclopedia of Software Engineering.)
- COINS 92-65, An Integrated Approach to Dynamic Task and Resource Management in Multiprocessor Real-Time Systems, C. Shen, (PhD Thesis).
- Well-Timed Scheduling: A Framework for Real-Time Scheduling, submitted to *IEEE Transactions on Parallel and Distributed Systems*, Dec. 1991, revised July 1992, G. Zlokapa, J.A. Stankovic, K. Ramamritham.
- Window MAC Protocols for Real-Time Communication Services, submitted to *IEEE Transactions on Communication*, Feb. 1991, revised July 1992.
- Architecture and OS Support for Predictable Real-Time Systems, D. Niehaus, E. N ahum, J. Stankovic, K. Ramamritham, March 1992.

4.4 Honors:

- Stankovic - Elected Fellow of the IEEE.
- Ramamritham - to Full Professor.

PI Names: Krithi Ramamritham, John A. Stankovic
PI Institution: University of Massachusetts
PI Phone Number: 413 545-0196 or 413 545-0720
PI E-mail Address: krithi@nirvan.cs.umass.edu, stankovic@cs.umass.edu
Grant or Contract Title: Predictable and Adaptable Complex Real-Time Systems
Grant or Contract Number: N00014-92-J-1048
Reporting Period: 1 Oct 92 - 30 Sep 93

5. Research Transitions and DoD Interactions:

Indirect Research Transitions:

Stankovic has been working with Advanced System Technologies, Inc. for the Naval Surface Warfare Center, White Oak Lab on establishing the basis for integrating real-time scheduling results from Spring and other research groups into a Knowledge Based DOS Assistant. A new contract in this area has been approved, but not yet begun.

Ramamritham has continued his collaborations with the researchers at AT&T Bell Labs, Murray Hill, on real-time concurrent C; working in ideas developed in Spring into that real-time language.

Our new tutorial text entitled, Advances in Real-Time Systems was published in September '93.

The Spring project is one of the focal points of the department's Center for Research in Real-Time Intelligent Complex Computer Systems (CRICCS). CRICCS houses the Center for Automated Real-Time Systems (CARTS). Stankovic is the director of CARTS; Ramamritham is an assistant director. As part of this center, interactions with several industrial participants have been initiated. We have ported the Spring kernel and associated tools to CARTS where flexible manufacturing based on high precision robots and vision systems is being investigated. Some of this work is funded by NSF and ARPA.

We have established a agreement with the Scuola Superiore S. Anna in Pisa, Italy for exchange of technology and personnel related to CARTS and the Spring technology.

We are hosting a visitor who is interested in real-time systems from Mitsubishi, Japan, for a year.

We are in the initial stages of filing for a patent on the Spring scheduling chip via ACSIOM, a non-profit company set up explicitly to transition technology to industry. This chip has large potential, not only for dynamic on-line real-time scheduling, but also for general scheduling problems.

Stankovic taught a week long intensive course on real-time computing, and developed a senior – first year graduate level course for UMASS (being taught this Fall). He also taught at the NATO institute on real-time computing.

Ramamritham taught a week long research and development oriented course on real-time systems at Alcatel Research Center in Vienna, Austria.

Ramamritham also taught a tutorial on real-time databases at (1) the International Conference on Management of Data (SIGMOD Annual Conference), Washington, D.C., and (2) Conference on Management of Data (COMAD), Bangalore, India.

Stankovic introduced the Spring technology into the Predictable Distributed Computing (PDCS) project in Europe; and as a result, a long-term collaboration has been initiated for dealing with adaptable fault tolerant real-time systems.

We have interacted with the Mechatronics community, establishing recognition that real-time issues will play a major role in mechatronics in the future.

Stankovic serves as the book series editor for real-time computing where five books have appeared during just the last year.

PI Names: Krithi Ramamritham, John A. Stankovic
PI Institution: University of Massachusetts
PI Phone Number: 413 545-0196 or 413 545-0720
PI E-mail Address: krithi@nirvan.cs.umass.edu, stankovic@cs.umass.edu
Grant or Contract Title: Predictable and Adaptable Complex Real-Time Systems
Grant or Contract Number: N00014-92-J-1048
Reporting Period: 1 Oct 92 - 30 Sep 93

6. Description of Software and Hardware Prototypes:

We developed the software simulation testbed for distributed real-time systems to include algorithms that perform scheduling with precedence and fault tolerance constraints. We further enhanced our Software Generation System which enables the specification of complex task requirements such as task groups and the subsequent creation of the Spring Kernel with this information embedded in the run time data structures.

We are also developing a real-time active database simulator to evaluate real-time transaction protocols that deal with concurrency control for complex objects, recovery of transactions, and triggered transactions.

A new version of the Spring Kernel that supports precedence related tasks has been developed and is being used with a robotics application to demonstrate flexible manufacturing techniques.

We have implemented a compiler that analyses real-time C programs for identifying potential blocking points and computes worst case execution time.